This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

Claim 1 (Currently Amended): A method for the controlled execution of a program, the program being intended for a virtual machine, on a portable data carrier, wherein

- the data carrier has a processor which executes at least a first and a second virtual machine,
- the program is executed both by the first and by the second virtual machine,
- the first and the second virtual machine both access a common heap in a non-volatile memory of the data carrier, wherein write operations to the common heap are only performed by one of the first and second virtual machines,
- an operating state of the first virtual machine and an operating state of the second virtual machine are checked during execution of the program for correspondence, and
- execution of the program is aborted if a difference is found between the operating state of the first virtual machine and the operating state of the second virtual machine.

Claim 2 (Previously Presented): A method according to claim 1, wherein checking of the operating state of the first virtual machine and of the operating state of the second virtual machine for correspondence comprises checking whether the state of a program counter of the first virtual machine is the same as the state of a program counter of the second virtual machine.

Claim 3 (Previously Presented): A method according to claim 1, wherein checking of the operating state of the first virtual machine and of the operating state of the second virtual machine for correspondence comprises checking whether the level of a stack pointer of the first virtual machine is the same as the level of a stack pointer of the second virtual machine.

Claim 4 (Previously Presented): A method according to claim 1, wherein checking of the operating state of the first virtual machine and the operating state of the second virtual machine for correspondence comprises checking whether a value of the most recent element in a stack associated with the first virtual machine is the same as a value of the most recent element in a stack associated with the second virtual machine.

Claim 5 (Previously Presented): A method according to claim 1, wherein checking of the operating state of the first virtual machine and of the operating state of the second virtual machine for correspondence is in each case performed after an instruction of the program has been executed both by the first and by the second virtual machine.

Claim 6 (Canceled).

Claim 7 (Currently Amended): A method according to ~~claim 6~~ claim 1, wherein, when an instruction of the program that contains a write operation to the common heap is being executed, a write operation is performed only by the first virtual machine.

Claim 8 (Previously Presented): A method according to claim 7, wherein the instruction of the program is executed first by the first virtual machine and then by the second virtual machine, and, instead of performing the write operation, the second virtual machine checks whether a value that is to be written is present in the heap at the location that is to be written to.

Claim 9 (Previously Presented): A method according to claim 1, wherein the program is a *Java Card Applet* intended for execution by a JCVM (*Java Card Virtual Machine*).

Claim 10 (Currently Amended): A portable data carrier, having a processor, a non-volatile memory, an operating system, at least a first and a second virtual machine, and a program, wherein

- the processor executes both the first and second virtual machine,
- the program is executed both by the first and by the second virtual machine,
- the first and the second virtual machine both access a common heap in the non-volatile memory of the data carrier, wherein write operations to the common heap are only performed by one of the first and second virtual machines,
- the operating system controls the processor to check the operating state of the first virtual machine and the operating state of the second virtual machine during execution of the program for correspondence, and
- the operating system controls the processor to abort execution of the program if a difference is found between the operating state of the first virtual machine and the operating state of the second virtual machine.

Claim 11 (Canceled).

Claim 12 (Previously Presented): A portable data carrier according to claim 10, wherein the data carrier is one of a chip card and a chip module.

Claim 13 (Previously Presented): A portable data carrier according to claim 10, wherein checking of the operating state of the first virtual machine and of the operating state of the second virtual machine for correspondence comprises checking whether the state of a program counter of the first virtual machine is the same as the state of a program counter of the second virtual machine.

Claim 14 (Previously Presented): A portable data carrier according to claim 10, wherein checking of the operating state of the first virtual machine and of the operating state of the second virtual machine for correspondence comprises checking whether the level of a stack pointer of the first virtual machine is the same as the level of a stack pointer of the second virtual machine.

Claim 15 (Previously Presented): A portable data carrier according to claim 10, wherein checking of the operating state of the first virtual machine and the operating state of the second virtual machine for correspondence comprises checking whether the value of the most recent element in a stack associated with the first virtual machine is the same as the value of the most recent element in a stack associated with the second virtual machine.

Claim 16 (Previously Presented): A portable data carrier according to claim 10, wherein checking of the operating state of the first virtual machine and of the operating state of the second virtual machine for correspondence is in each case performed after an instruction of the program has been executed both by the first and by the second virtual machine.

Claim 17 (Canceled).

Claim 18 (Currently Amended): A portable data carrier according to ~~claim 17~~ claim 10, wherein, when an instruction of the program that contains a write operation to the common heap is being executed, the write operation is performed only by the first virtual machine.

Claim 19 (Previously Presented): A portable data carrier according to claim 18, wherein the instruction of the program is executed first by the first virtual machine and then by the second virtual machine, and, instead of performing the write operation, the second virtual machine checks whether the value that is to be written is present in the heap at the location that is to be written to.

Claim 20 (Previously Presented): A portable data carrier according to claim 10, wherein the program is a *Java Card Applet* intended for execution by a JCVM (*Java Card Virtual Machine*).

Claim 21 (Currently Amended): A tangible computer program product having program instructions for causing a processor of a portable data carrier to perform a method for the controlled execution of a program, the program being intended for a virtual machine, wherein

- the processor executes at least a first and a second virtual machine,
- the program is executed both by the first and by the second virtual machine,
- the first and the second virtual machine both access a common heap in a non-volatile memory of the data carrier, wherein write operations to the common heap are only performed by one of the fits and second virtual machines,
- the operating state of the first virtual machine and the operating state of the second virtual machine are checked during execution of the program for correspondence, and
- execution of the program is aborted if a difference is found between the operating state of the first virtual machine and the operating state of the second virtual machine.

Claim 22 (Canceled).

Claim 23 (Currently Amended): A computer program product according to ~~claim 22~~ claim 21, wherein, when an instruction of the program that contains a write operation to the common heap is being executed, the write operation is performed only by the first virtual machine.

Claim 24 (Previously Presented): A computer program product according to claim 23, wherein the instruction of the program is executed first by the first virtual machine and then by the second virtual machine, and, instead of performing the write operation, the second virtual machine checks whether the value that is to be written is present in the heap at the location that is to be written to.

Claim 25 (Previously Presented): A computer program product according to claim 21, wherein the program is a *Java Card Applet* intended for execution by a JCVM (*Java Card Virtual Machine*).